Mapping Algorithms

Matthias Zytnicki
MIAT INRA
Outline

1. Introduction

2. Genome mapping
   - Hashing-based tools
   - Suffix array-based tools

3. Transcriptome mapping

4. Conclusion
1. Introduction

2. Genome mapping
   - Hashing-based tools
   - Suffix array-based tools

3. Transcriptome mapping

4. Conclusion
What is mapping

Desired definition
Map a read: predict the *locus* from which the read originates.
What is mapping

**Desired definition**
Map a read: \textit{predict} the \textit{locus} from which the read originates.

**Data**

- genome
- read

\[ \text{mapping} \rightarrow \text{genomic coordinate(s)} \]

**Example**

- `@SEQ_ID GATTT + ccedd`

\[ \text{BWA} \rightarrow @SEQ_ID 113 1\ldots \]

**Assumption**
A read is likely to map at a locus \textit{iff} similarity is high.
What is mapping

Desired definition
Map a read: *predict* the *locus* from which the read originates.

Data
- genome read → mapping → genomic coordinate(s)

Example
- @SEQ_ID
- GATTT +
- ccedd → BWA → @SEQ_ID 113 1...

Assumption
A read is likely to map at a locus iff similarity is high.

Implemented definition
Map a read: *list* the *loci* with less than *k* errors.
First implications

Ambiguity in the mapping

Mapping $\neq$ Alignment

genome

read

- ACGTACGT
- ACGTAGT-

is a valid mapping with 2 mismatches.
First implications

Ambiguity in the mapping

Mapping \( \neq \) Alignment

\[ \text{genome} \]

\[ \text{read} \]

\[ \text{genome} \]

\[ \text{read} \]

\[ \text{mapping} \]

\[ \text{valid mapping with 2 mismatches} \]
First implications

Ambiguity in the mapping

Mapping 
̸

̸= Alignment

ACGTACGT

ACGTAGT-

is a valid mapping with 2 mismatches.
First implications

Ambiguity in the mapping

Mapping $\neq$ Alignment

genome

read

is a valid mapping with 2 mismatches.
First implications

Ambiguity in the mapping

Mapping $\neq$ Alignment

<table>
<thead>
<tr>
<th>genome</th>
<th>ACGTACGT</th>
<th>is a valid mapping with 2 mismatches.</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>ACGTAGT-</td>
<td></td>
</tr>
</tbody>
</table>
Tools...
Outline

1 Introduction

2 Genome mapping
   Hashing-based tools
   Suffix array-based tools

3 Transcriptome mapping

4 Conclusion
Outline

1. Introduction

2. Genome mapping
   - Hashing-based tools
   - Suffix array-based tools

3. Transcriptome mapping

4. Conclusion
Seed-and-extend algorithm

Idea

1. Get the $k$-mers of the genome,

Example

Genome is AACGTAC, read is CCGT

\[ \text{AACGTA} = \text{AAC ACG CGT GTA} \]
Seed-and-extend algorithm

Idea

1. Get the $k$-mers of the genome,
2. “Sort” them,

Example

Genome is AACGTAC, read is CCGT

AAC
ACG
CGT
GTA
Seed-and-extend algorithm

Idea

1. Get the \( k \)-mers of the genome,
2. “Sort” them,
3. Get the \( k \)-mers of a read,

Example

Genome is AACGTAC, read is CCGT

\[
\text{CCGT} = \text{CCG} \; \text{CGT}
\]
Seed-and-extend algorithm

Idea

1. Get the $k$-mers of the genome,
2. “Sort” them,
3. Get the $k$-mers of a read,
4. Compare the two,

Example

Genome is AACGTAC, read is CCGT

$\text{CCGT} = \text{CCG} \quad \text{CGT}$

\[ \begin{array}{c}
\text{AAC} \\
\text{ACG} \\
\text{CGT} \\
\text{GTA}
\end{array} \]
Seed-and-extend algorithm

Idea

1. Get the $k$-mers of the genome,
2. “Sort” them,
3. Get the $k$-mers of a read,
4. Compare the two,
5. Finish the alignment.

Example

Genome is AACGTAC, read is CCGT

\[
\begin{array}{cccccc}
A & A & C & G & T & A \\
C & C & G & T & & \\
\end{array}
\]
Step 2: Extension with errors

Needleman–Wunsch

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>→ 1</td>
<td>→ 2</td>
<td>→ 3</td>
<td>→ 4</td>
</tr>
<tr>
<td>↓</td>
<td>↘</td>
<td>↘</td>
<td>↘</td>
<td>↘</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>↓</td>
<td>↘</td>
<td>↘</td>
<td>↘</td>
<td>↘</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>↓</td>
<td>↘</td>
<td>↘</td>
<td>↘</td>
<td>↘</td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Optimizations

Banded Needleman–Wunsch, FSA, parallel Shift-OR, vectorization, SIMD, . . .
Choice of $k$ — Example 1

Example

Consider sequence ACGCGTGTA and read ACACGAGTA.

- With $k = 3$, seed GTA matches.
- With $k = 4$, no seed match.
Choice of $k$ — Example 1

Example
Consider sequence ACGCGTGTGA and read ACACGAGTA.

- With $k = 3$, seed GTA matches.
- With $k = 4$, no seed match.

Pigeon hole principle
With $r$ errors, $r + 1$ seeds.
⇒ small $k =$ more sensitive.
Example

Consider sequence ACGACGACGACGACGT and read ACGT.

$k = 3$:

- ACG → 0, 3, 6, 9, 12
- CGA → 1, 4, 7, 10
- CGT → 13
- GAC → 2, 5, 8, 11

⇒ 4 matches, 1 extension succeeds.

$k = 4$:

- ACGA → 0, 3, 6, 9
- ACGT → 12
- CGAC → 1, 4, 7, 10
- GACG → 2, 5, 8, 11

⇒ 1 matches, 1 extension succeeds.

Remark: $k$-mers with many occurrences usually are discarded.
Example

Consider sequence ACGACGACGACGACGT and read ACGT.

$k = 3$:
- ACG → 0, 3, 6, 9, 12
- CGA → 1, 4, 7, 10
- CGT → 13
- GAC → 2, 5, 8, 11
⇒ 4 matches, 1 extension succeeds.

$k = 4$:
- ACGA → 0, 3, 6, 9
- ACGT → 12
- CGAC → 1, 4, 7, 10
- GACG → 2, 5, 8, 11
⇒ 1 matches, 1 extension succeeds.

Remark

$k$-mers with many occurrences usually are discarded.
**Conclusion so far**

**Trade-off**

- Small $k$: more sensitive, slower.
- High $k$: more specific, larger database (size: up to $4^k$).
Conclusion so far

Trade-off

- Small $k$: more sensitive, slower.
- High $k$: more specific, larger database (size: up to $4^k$).

Tentative complexity

- Pre-process genome (done once).
- Cut a read into $k$-mers (fast).
- Map each $k$-mer to the table (fast for each $k$-mer).
- Consider every position in the genome (variable).
- Extend with errors (slow).
Conclusion so far

Trade-off

• Small $k$: more sensitive, slower.
• High $k$: more specific, larger database (size: up to $4^k$).

Tentative complexity

• Pre-process genome (done once).
• Cut a read into $k$-mers (fast).
• Map each $k$-mer to the table (fast for each $k$-mer).
• Consider every position in the genome (variable).
• Extend with errors (slow).

Drawbacks

Slow or not sensitive when:
• accepting many errors,
• accepting highly repeated seeds.
Outline

1 Introduction

2 Genome mapping
   Hashing-based tools
   Suffix array-based tools

3 Transcriptome mapping

4 Conclusion
GATTACA is:
Looking for a read

Idea
Simply follow the right arrow.

Example
Look for TAC.
Looking for a read

Idea
Simply follow the right arrow.

Example
Look for TAC.
Look for TAT.
Looking for a read

Idea
Simply follow the right arrow.

Example
Look for TAC.
Look for TAT.

Problem
*Suffix trees do not fit in memory.*

*Use suffix arrays instead:*

- same algorithms,
- condensed data structure.
Handling errors

Example
Look for GAC with 1 error.
Handling errors

Example
Look for GAC with 1 error.
G
Example
Look for GAC with 1 error.
GA
Handling errors

Example

Look for GAC with 1 error.
GAC
Handling errors

Example
Look for GAC with 1 error.

Problem
The search space is very large!
Conclusion so far

Tentative complexity

- Pre-process the genome (done once).
- Look for a read (quite fast if no error).

Slow or not sensitive when:
- accepting many errors (small seed).
Outline

1. Introduction

2. Genome mapping
   - Hashing-based tools
   - Suffix array-based tools

3. Transcriptome mapping

4. Conclusion
RNA mapping $\neq$ DNA mapping
Difficult cases

- many differences (mutations and/or sequencing errors),
- repeated sequence,
- read on 3+ exons,
- gene or pseudogene?

(Kim et al., Gen. Biol., 2013)

- end of read on another exon,

Incorrect mapping (non-gapped alignment)

Correct mapping (spliced alignment)

- read on unknown and poorly expressed junction.
Tophat1

(Map reads to whole genome with Bowtie)

Collect initially unmappable reads

Assemble consensus of covered regions

Generate possible splices between neighboring exons

Build seed table index from unmappable reads

Map reads to possible splices via seed-and-extend

(Trapnell et al., Bioinformatics, 2009)

- uses Bowtie,
- problems for pseudogenes.
Algorithms

(a) Exon-first approach

Exon read mapping

Spliced read mapping

(b) Seed-extend approach

Seed matching

Seed extend

(Garber et al., Nat. meth., 2011)
Tophat2

(Kim et al., Gen. Biol., 2013)
Outline

1. Introduction

2. Genome mapping
   - Hashing-based tools
   - Suffix array-based tools

3. Transcriptome mapping

4. Conclusion
Other considerations

**Quality**  Current tools provide read quality. Better use it.

**Pair end reads**  Most tools have a special “fragment rescue” mode.

**Low quality 3’ ends**  Some tools truncate them.

**Binary encoding**  A: 00, C: 01, G: 10, T: 11, other: ???
Tool comparison

- **Sensitivity**: map most reads,
- **Specificity**: mappings are correct,
- **Time**
- **Memory**

⇒ Balance between the criteria.
## DNA mapping tools

<table>
<thead>
<tr>
<th>name</th>
<th>seed-and-extend</th>
<th>pigeon hole</th>
<th>spaced seed</th>
<th>q-gram</th>
<th>suf. tree</th>
<th>B.–W. notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSAHA</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blat</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Introns</td>
</tr>
<tr>
<td>MUMmer2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eland</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAQ</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOAP</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PatMaN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMAP</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SeqMap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Several seeds</td>
</tr>
<tr>
<td>Mosaik</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZOOM</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>Minimal # seeds</td>
</tr>
<tr>
<td>PASS</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No heuristic</td>
</tr>
<tr>
<td>SOAP2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BWA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SNP detection</td>
</tr>
<tr>
<td>SHRiMP</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>Mapping quality</td>
</tr>
<tr>
<td>Bowtie</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Color space</td>
</tr>
<tr>
<td>mrFAST</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Trade-off between speed and accuracy</td>
</tr>
<tr>
<td>RazerS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gives all hits</td>
</tr>
<tr>
<td>CloudBurst</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MapReduce</td>
</tr>
<tr>
<td>MPScan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No error</td>
</tr>
<tr>
<td>PerM</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BFAST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2-level hash table</td>
</tr>
<tr>
<td>Segemehl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GNUMap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>Cache-oblivious</td>
</tr>
<tr>
<td>mrsFAST</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>novoalign</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>GASSST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stampy</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rNA</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SOAP3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X Uses GPU</td>
</tr>
<tr>
<td>Bowtie2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X SIMD</td>
</tr>
<tr>
<td>Saruman</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>GPU</td>
</tr>
<tr>
<td>BarraCUDA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X GPGPU</td>
</tr>
</tbody>
</table>
BWA vs Bowtie2

(Li Website: http://lh3lh3.users.sourceforge.net/alignROC.shtml)
Bowtie2 vs BWA

(Langmead et al., Nat. Meth., 2012)
“Unbiased” comparisons

• Matthew Ruffalo, Thomas LaFramboise, Mehmet Koyutürk, *Comparative analysis of algorithms for next-generation sequencing read alignment*, Bioinformatics (2011), vol 27, pp. 2790–2796.


⇒ Results are already irrelevant!
RGASP

The RNA-seq Genome Annotation Assessment Project.

(Engström et al., Nat. Meth., 2013)
Comparison transcript assembly.

(Engström et al., Nat. Meth., 2013)
Which tool should I use?

Common situations
You can use widely-used tools.

- **DNA-Seq**: BWA(-SW/-mem), Bowtie2
- **RNA-Seq**: TopHat2, Star

- Cannot refuse your paper!
- Well maintained.
- Large software suite (Tuxedo).

A niche?

- mrFAST: for high number of copies
- SHRiMP: for color space
- Stampy: for highly divergent sequencing (with BWA)
- CloudBurst: with Map/Reduce
- SOAP3: for GPU
Which tool should I use?

Common situations
You can use widely-used tools.

- DNA-Seq BWA(-SW/-mem), Bowtie2
- RNA-Seq TopHat2, Star

- Cannot refuse your paper!
- Well maintained.
- Large software suite (Tuxedo).

Want to buy France?
Use CRAC.

A niche?

- mrFAST: for high number of copies
- SHRiMP: for color space
- Stampy: for highly divergent sequencing (with BWA)
- CloudBurst: with Map/Reduce
- SOAP3: for GPU